Kaggle - GettingStarted prediction Competition Digit Recognizer (MNIST)

Steve Levesque
https://stevelevesque.dev
hello@stevelevesque.dev

Abstract

The digit recognition task in machine learning or data science is the "hello world" of the domain. It is a good point to start up in the field. This article will describe the steps used to achieve "acceptable" results (by this statement, a score around 99%, and lower than 99.6%), possible ways to increase it fairly in terms of baises and cheating, and discard unreasonably high scores such as 99.957% or worst like 100%. The redaction's content difficulty is made to be light and made for general public, for those who could graspe that there are a lot of superficial explainations in term of results and advances.

1 Introduction

For this type of task, it is easily possible (especially as for today, 2021-22) to build a model that can reach above 90% in a matter of minutes, or around 97-99% in less than an hour. The tricky part about MNIST digit recognition is mainly for the last hundred digits or so, as for the model AND humans. Seeing by ourselves such writings is hard even for us to classify since there are very badly written number like a four that ressembles a lot to a nine and others of the sort.

2 MNIST Dataset

2.1 Data Description

The MNIST dataset consists of numbers between 0 and 9 that are handwritten. The traditionnal task with the digit is to classify each picture of 784 pixels or of dimension 28×28 with 1 dimension of color (black and white) to its rightful category (a number between 0 and 9 respectively).

2.2 Data Augmentation

To help generalize in machine learning, it is a good solution to have more data that is not the test set nor a subset/superset of it. (to avoid bias or at a higher pitch, cheating) It is possible to modify the images of our train set with small equivariant variations (invariant variations, as the word would suggest, is just copying the image and doubling the size of our training set, which is useless) such as translation, rotation and zooming (i.e. not as a whole 180 degrees, else 6 and 9 would be the same).

With the augmentations, digits will have more chances to be righfully classified if someone writes with a more italic tone or with a unsteady pattern. We can see in the first 9 digits of the test set of Figure 1 that there are four zeros written with a different size, shape, marker size, etc.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

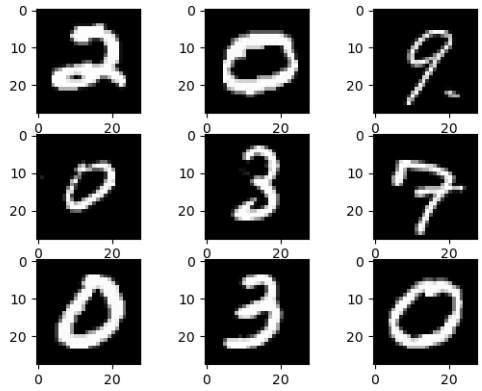

Figure 1: First 9 digits of the Kaggle MNIST test set

3 Algorithms

3.1 XGBoost

"XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable." The first sentence from their website describes well XGBoost in a few words. It can train and predict with a score quite acceptable for the little ammount of time it takes it to solve huge diverse problems.

The score is 93.78% for about 10 minutes of training.

3.2 KNN

Another simple and easy algorithm to use is K Nearest Neighbors. It works well with a lot of problems that takes distances into account. We can use the number of neighbors equal to one to have a medium decent result. It won't be the best, but with how MNIST is with distances towards each digit that are normally close to eachother if they are of the same group, it will classify all even looking numbers together without hassle. The score won't be 99%, since there will be errors when a seven looks like a one and all similar irregularities.

The score is 97.01% for about 30 minutes of training.

3.3 CNN

However, CNN is of choice if the intent is to perform well on a task that requires comupter vision such as this one. We start by defining a conv-net for our model. Keras is used for the network, data augmenting, model saving/loading and finally for prediction. Next, we choose the right number of epochs and batch size for the training. When done, the final step is plotting and comparing the results. It is important to see what is gain and for how much (i.e. time spent for every 0.1% gained).

The results are made with data augmentation beforehand. With only 2 epochs and a large batch size of 128, we can reach as high as 98.69%. This is good taking into account that it takes around 10 minutes to train. With 50 epochs and 64 batch size, we can reach up to 99.55%. With double the ammount of epochs (100), we only get 0.021% increase. Thus around double the time for only that much more.

The first score is (98.69%) for about 10 minutes of training.

The second (99.55%) for about 30 minutes of training.

The third (99.57%) for about 1 hour of training.

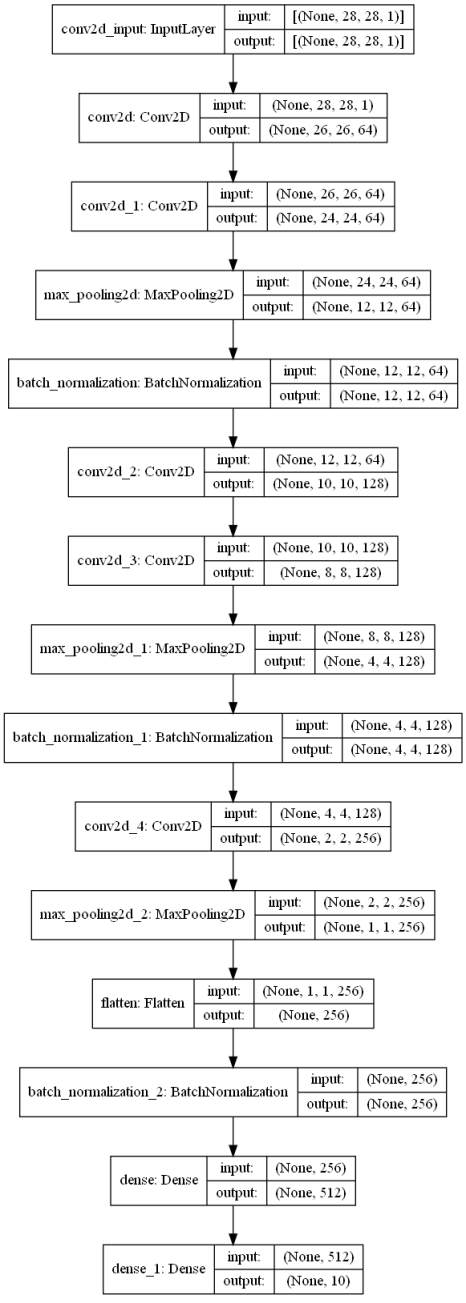

Figure 2: Conv-net with Keras used for the model.

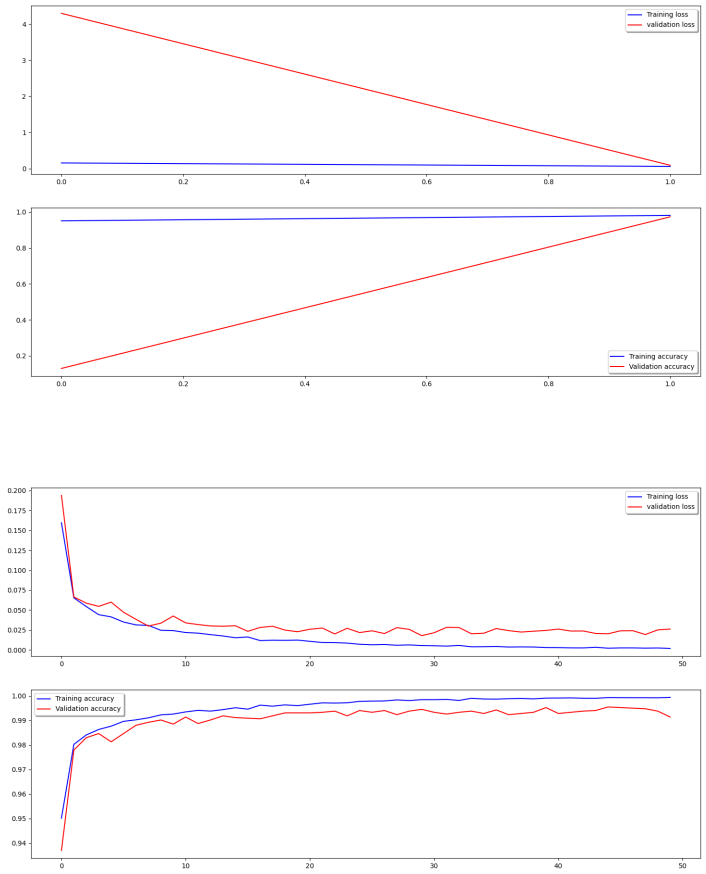

Figure 3: Loss and accuracy for 2-50 epochs and 128-64 respectively.

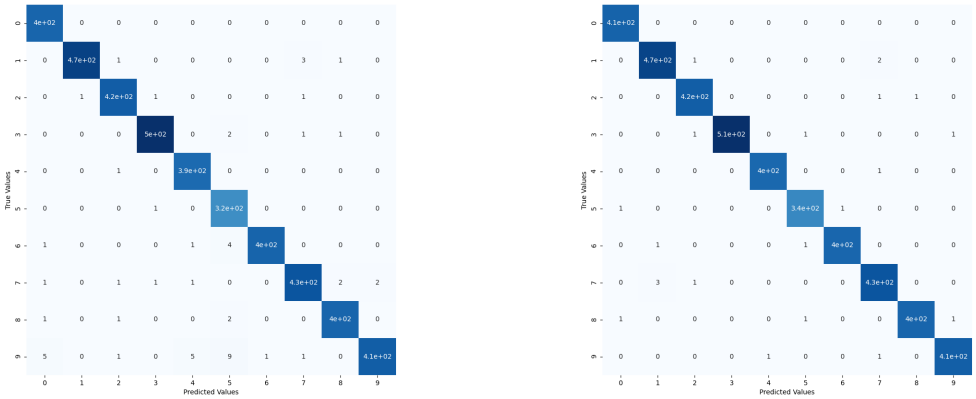

Figure 4: Confusion matrix for 2-50 epochs and 128-64 respectively.

4 Results

The table below contains results obtained from the algorithms above and some other less legitimate techniques. Scores aboves 99.57% will be discused on following sections. Above this treshold for the last digits more complicated to classify, highter accuracy is always possible, but with high consumtion of time and/or resources.

NB: The public score constitutes the private as well, since it is 100% the data for learning purposes.

Table 1: Leaderboard Results by Algortihm

Algorithms	Public
XGBoost	∼93.78%
KNN (neighbours = 1)	∼97.01%
CNN 2 epochs and 128 batch-size	∼98.69%
CNN 50 epochs and 64 batch-size	∼99.55%
CNN 100 epochs and 64 batch-size	∼**99.57%**
CNN 50 epochs and 64 batch-size with QMNIST	∼99.96%
1 for-loop on MNIST original dataset	100%

5 Biased Methods

There are lots attractive results like the 99.96% with QMNIST or the infamous 100% that is nothing but a scam, especially with such a problem where even humans can't reach 100% unless lucky because the last 1% of the digits are so badly written we have a hard time trying to figure them out.

"The QMNIST dataset was generated from the original data found in the NIST Special Database 19 with the goal to match the MNIST preprocessing as closely as possible."[1],

How is it possible to get such results? The answer resolves around a simple fact, the labels are given since the test set is a subset, directly or not of the train set. A result of 100% can be achieved by brute force on the MNIST original dataset which contains every elments of the test set on the Kaggle competition. The same principle applies for QMNIST since even though it is not a perfect subset, it is indirectly by the fact that the generation as for goal to mimic MNIST. With 120k digits from a reconstruction heavily based over MNIST with around 10 digits over 60k of them are invariant counterparts of the test set, we can assume that every other equivariant digits are very similar to MNIST and that in consequence it is possible the test set is a subset of QMNIST. With a score of 99.96%, it could be a good sign that it is a subset.

6 Possible Derivations

Is it the end? No, and it is possible to find articles and works that can perform better than 99.57% without any use of the test set and super/sub sets of anything related.

6.1 Ensemble on Trained Models

As used in this paper[2], it would be possible to use multiple different models and achieve a result close to 99.8% and 99.9% (the paper stipulates an accuracy of 99.91% on MNIST).

There is also this ensemblist method used on Kaggle to obtain 99.76%[3] that uses a similar ensemble technique.

[1] https://github.com/facebookresearch/qmnist

[2] https://arxiv.org/pdf/2008.10400v2.pdf

[3] https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist/notebook

6.2 Ensemble on Models and Algorithms

There could be a way to have a better berformance than humans if we ensemble multiples algortihms which has multiple models. This could reach a point where the machine can learn the pattern used for "generating" the MNIST digits and classify correctly the hundred last incomprehensible digits left.

Plus, it could exploit other aspects not directly linked to a CNN with ensemble models, such as distances to give possible additional insights.

However, this would cost an exponentional amount of resources and time for only about a 0.01% increase.

7 Conclusion

As of the date this has been written, there is a treshold where a CNN model can go and where ensemblist methods with high data augmentations are necessary to scrape off an additional fraction of a percent or two. Everything reaching 99.9% or 100% should be discarted, as a human would have difficulty to reach such precision.

References

Kaggle Competition Digit Recognizer https://www.kaggle.com/c/digit-recognizer/overview

Hyperopt http://hyperopt.github.io/hyperopt/

scikit-learn https://scikit-learn.org/stable/

XGBoost https://xgboost.readthedocs.io/en/stable/

QMNIST https://github.com/facebookresearch/qmnist

Confusion Matrix Plot https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Matplotlib.pyplot https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html

Python https://www.python.org/

Keras https://keras.io/

Tensorflow https://www.tensorflow.org/

High Scoring MNIST Solutions https://paperswithcode.com/sota/image-classification-on-mnist

Paper - An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition https://arxiv.org/pdf/2008.10400v2.pdf

Kaggle MNIST 0.99757 https://www.kaggle.com/cdeotte/25-million-images-0-99757-mnist/notebook